Working in Teams

The simple fact of the matter is that it is unlikely that you will be a “Lone Ranger” in the Computing industry.

A Personal View

“I would much rather work on my own than with other people. I find that other people are much more likely to get in the way and slow me down. Mostly people are irritating when working in a team.”

The nearest you are ever going to get to being a Lone Ranger is if you are a self employed sole trader.

However you will still have customers that need to be interacted with involving clear communication. You will still need to interact with other people at various stages of development.

In industry the odds are that you will never have a full system to develop on your own.

What happens is that you are allocated sections of a system.

This is exactly what we are imitating in this module. You are all working on the larger Project Bank but each one of you is in charge of a sub section of that system.

This creates a number of problems that need to be resolved.

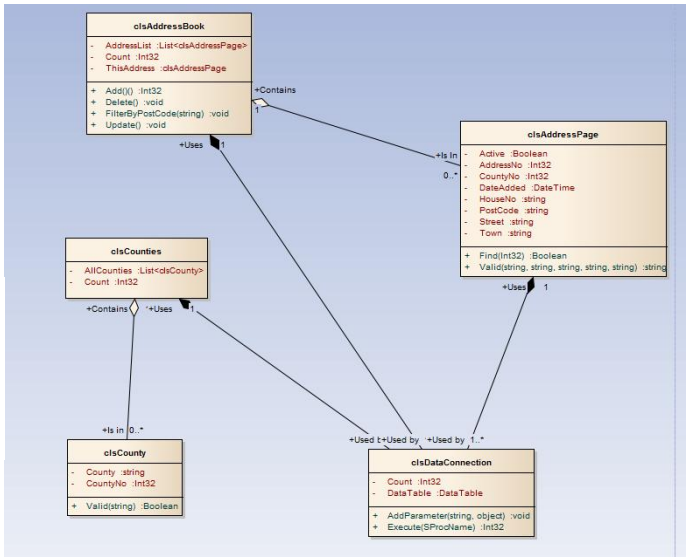
Version Control

One of the problems you are going to face is how to manage version control.

What I mean is this.

Look at the following class diagram…

Bob



Jane

Let's assume that there are two developers working on the above system. (Jane and Bob)

Jane is working on the address book (clsAddressPage & clsAddressBook) Bob is working on the classes clsCounties and clsCounty.

If Bob wishes to extend the rather limited functionality of clsCounties to now include Add, Delete and Update what impact will this have on Jane?

The first question is how are they sharing their code?

If the project you are working on is stored on a single computer and each developer has exclusive access to the machine one at a time there isn't a problem.

Bob…
Makes changes to the code for Counties
Saves the work to the computer
Goes off and does something else

Jane…
Makes changes to the address book code
Has access to the code Bob modified above
No problems with version control

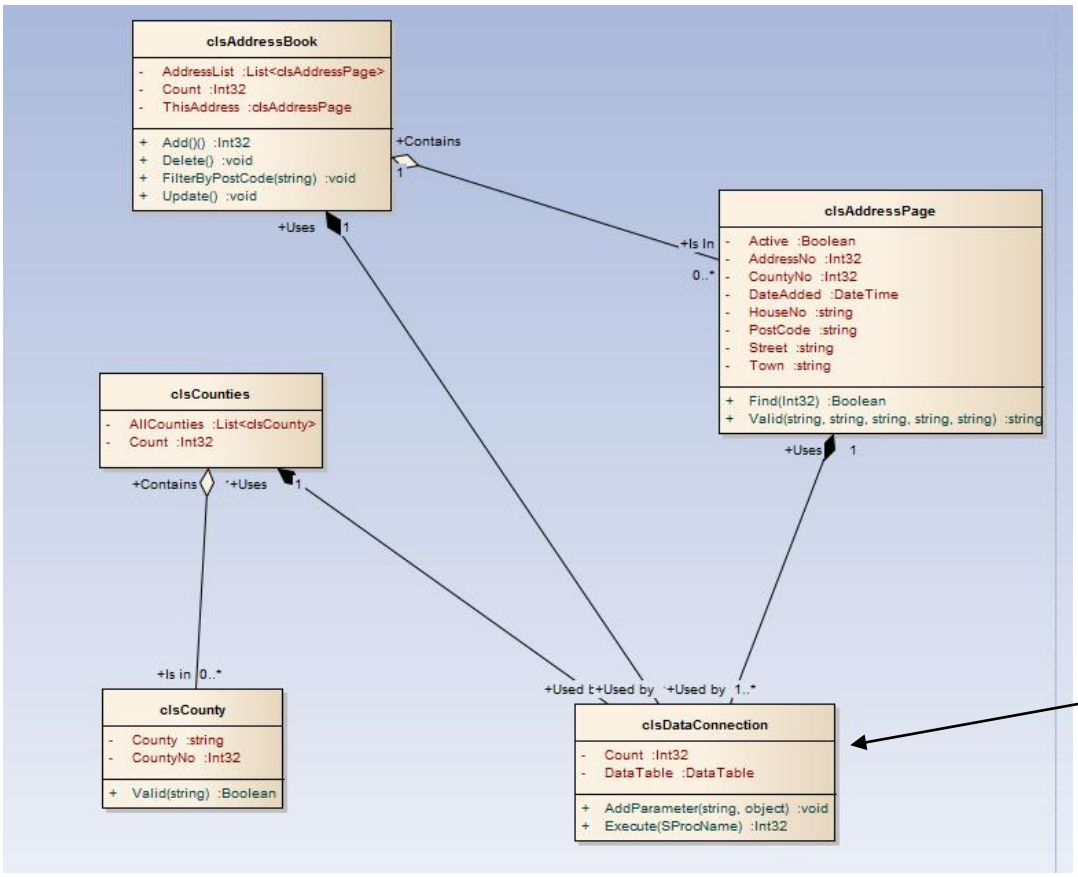
Version control becomes a problem as systems start to become distributed across networks.

	
Bob	Jane

What is much more realistic is that on a network Bob and Jane are going to make changes to their allocated parts of the system simultaneously.

Bob and Jane come into work at 9am with a deadline of 12pm for the system being delivered to the client.

What happens if Bob and Jane have to make changes to the shared data connection to the database?



Jane opens clsDataConnection in Visual Studio
Bob adds a new method to clsDataConnection
Bob saves his work on the class
Jane completes her work modifying the code for clsAddressBook
In the process Jane also saves clsDataConnection
This reverts clsDataConnection back to its original version overwriting the new code created by Bob

If Bob makes changes to the counties code, Jane makes changes to the address book code, both save their work.

If the code is hosted on a central server this is not going to be too big a problem as they are still essentially working from the same computer (the server).

However what if (as is becoming more common today) they are both working on the code off site?

This time there are local copies of the system code across three machines. (The two off site machines and the central server)

How do we set about synchronising the three copies of the system such that both Bob and Jane have access to the most up to date version of the code?

These are a few of the technical problems that need to be addressed when working on a system as a team. This is typically referred to as version control.

Version control has wider implications from the point of view of the customer too.

Bob and Jane work on the system to produce version 1.0 of the product
Version 1 is released to customers
More work is carried out and version 1.1 is released which is chargeable
Some customers report that with version 1.0 there is a bug
Customers refuse to upgrade to version 1.1 due to the cost and want a fix for version 1.0 (This will create version 1.01)
The question then arises how to manage the code not just to handle multiple versions of the code in use by developers but also multiple versions of the code over the history of the system.

Version Control is a complex and important topic in software development.

It is not just an issue with the code for the system but all documentation along the way. We need mechanisms for sharing design documentation such that all changes are synchronised and logged throughout the history of the system.

Communication

Communication is by far THE most important issue to address in working as a team.

The customer is always the first priority in communication.

Since we are lacking customers in this module we will all play the role of "customer".

In the lab this week you will all look at each other's prototype and make recommendations for additional / missing functionality.

The Desire Path

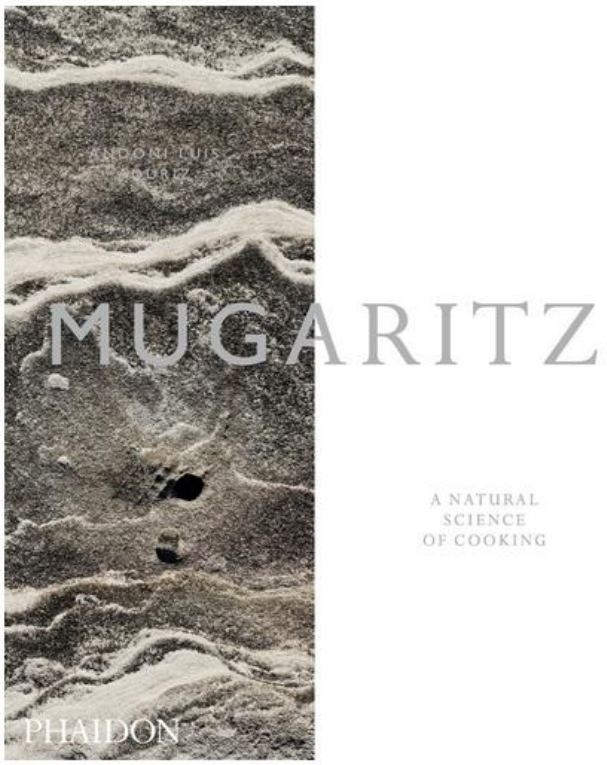
What are we looking at here?

This is an example of what is called the desire path. You often see it in parks.

What happens is that the designers of a park set out the paths and walkways in the park only to find that people completely ignore the defined paths and make their own paths in the grass.

The fact is that as developers we will have our own ideas about how our system is going to work but if that doesn't match with what the customer wants then the system is going to fail.

Mugaritz Restaurant



The Mugaritz restaurant in Spain is considered one of the finest restaurants in the world.

Apparently one of the secrets of its success is that the meal is built around the customer.

In most restaurants when you sit at the table there is a glass of wine and cutlery set out assuming that you are going to eat a certain set of course and drink certain drinks.

In the Mugaritz when you sit at the table it is a blank slate. There isn't even a menu.

The staff will then ask you what you like to get a sense of your likes and dislikes. They will then put together a meal based on this information using the finest produce that they have in at that time.

Your meal is then uniquely prepared to fit in with your own personal tastes.

This level of tailoring is something we really need to keep in mind when creating software for customers. If we don't communicate with them then we will fail to meet the requirements.

Team Communication

You need to appreciate that when you make a change to a section of the system you are not changing something that is owned just by you but is owned collectively by the entire team.

Break the code that belongs to you and you break the code that belongs to the team!

There are number of low tech approaches to improving communication.

Have regular meetings with the team
Be aware of the members of the team your changes will impact on
Establish work practices that allow team members to know who is doing what when

We will look in a moment about what measures may be put in place to improve communication.

Documentation

Over the course of this module we have looked at a range of documentary artefacts that allow us to think about how the system is going to work but more importantly leave a paper trail of how we think the system is going to work.

- Detailed Written Specification
- Event Tables
- Use Case Diagrams
- Use Case Descriptions
- Early Prototype
- Test Plans
- Class Diagrams
- Sequence Diagrams
- Entity Relationship Diagrams

Be clear we have only scratched the surface of the documents potentially available in the real world.

The documentation needs to be clear and concise and more importantly it needs to be aligned with reality at all points.

For example…

- The class diagram needs to match the classes built within the system.
- The test plans must match with the tests in the test framework.
- The use cases need to be aligned with the way that the system actually works.

If there is a lack of alignment then it is a bit like using an old map to navigate a brand new road system.

Work Practices

For this module we will be adopting many of the practices of Agile/Extreme Programming (XP).

Scrum Development

Scrum development is a modern approach to developing systems that works on the assumption that a customer will change their mind as the system is created.

There are certain practices and roles in this process.

Product Backlog

The product backlog is a to-do list of what needs completing to deliver the system. Each member of the team needs to claim items off the backlog over the period of development.

In the case of scrum team members may be allocated dynamically to functions. Somebody may work on a function for a limited time and then be switched to another function based on specialism.

Items on the product backlog should be ordered by priority. The team needs to focus on high priority functionality first moving to less important system functions.

You must bring a copy of the backlog with you to future team meetings.

Scrum Master

Somebody in your team needs to take charge of the process to make sure that it is being implemented properly. They will also need to take charge of the product backlog. (More on that next)

Sprint

A sprint is a period of time where the team gets on with its agreed tasks. In this case a sprint will be two weeks. A sprint starts with a sprint planning meeting. At the planning meeting the product backlog needs to be reviewed and tasks allocated to the members of the team. After the two week sprint the team will review the work completed so far. This review process will involve reviewing the Product Backlog. Removing functions that are completed and reviewing the list ensuring it matches user requirements. During the sprint the team must communicate and record the meetings.

Pair Programming

Pair programming is a practice in XP where two programmers work on a task together.

One person in the pair takes the role of the code writer and the other takes the role of a code thinker.

Pair programming helps to spread the programming skill through the team.

Programming Pairs are again dynamically allocated over the life time of the project. Do not pair permanently with the same person and alternate the role of programmer and thinker.

Systems to Support the Team

Visual Studio
Team Foundation Server
Enterprise Architect
MS Project
Drop box / OneDrive

Where Next for the Module?

This week in the lab we will review the prototypes to see how complete they are. And reallocate the work load across the teams.

The lab this week will by only one hour long.

There will be no claims for credit in the lab. This is an opportunity to "fall back and re-group" and think about your first sprint, creating an initial product backlog.

We will hopefully identify parts of your system that are lacking along with the re-allocation of work such that you feel confident in completing the development of the system such that it meets the requirements.

As a team I want you to bring to the lab the assembled prototype you have handed in.

Your team needs to download a copy of the word document called Product Backlog from the module web site.

In advance of the lab you need to complete the date and team members. You will also need to complete the section Functions Identified. This will take the form of a bullet point list of what functions you have already thought about in your prototype design.

In the lab you will set up your team's prototype on a single machine.

Each team will spend around 10 minutes looking at your prototype. They will add to the word document under the heading "Functions Outstanding". Again this will take the form of a bullet point list.

At the end of the session you need to print out your product backlog and hand it in to your tutor.

You will be expected to produce an updated product backlog to each team meeting for the rest of the year.